

COPY

Attorney Docket No.: 16869c-016800US
HAL-ID 152

**PATENT APPLICATION
LOAD DISTRIBUTION OF MULTIPLE DISKS**

Inventor(s):

Akira Yamamoto, a U.S. citizen, residing at 1202 Ruppell Place, Cupertino
CA 95014

Takao Satoh, a citizen of Japan, residing at 372-6 Kamonomiya, Odawara-shi,
Kanagawa, Japan 250-0874

Shigeo Honma, a citizen of Japan, residing at 846-7 Kamonomiya, Odawara-
shi, Kanagawa, Japan 250-0874

Yoshihiro Asaka, a citizen of Japan, residing at 6-15 Nakazone, Odawara-shi,
Kanagawa, Japan 250-0866

Yoshiaki Kuwahara, a citizen of Japan, residing at 3731 Sobi, Odawara-shi,
Kanagawa, Japan 250-0851

Hiroyuki Kitajima, a citizen of Japan, residing at 63-14 Wakakusadai, Aoba-
ku, Yokohama-shi, Kanagawa, Japan 227-0045

Assignee:

Hitachi, Ltd.
6, Kanda Surugadai 4-Chome
Chiyoda-Ku
Tokyo, Japan 101-8010

Entity: Large

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
Tel: 415-576-0200

LOAD DISTRIBUTION OF MULTIPLE DISKS

CROSS-REFERENCES TO RELATED APPLICATIONS

This is a continuation-in-part application of U.S. Application No. 09/619,000,
5 filed July 18, 2000, which is a continuation of Application No. 09/116,344, filed July 16,
1998, now U.S. Pat. No. 6,108,750, which is a continuation of Application No. 08/868/075,
filed June 3, 1997, now U.S. Pat. No. 5,835,938, which is a continuation of Application No.
08/355,274, filed December 12, 1994, now U.S. Pat. No. 5,680,574, which is a continuation
of Application No. 07/648,998, filed January 31, 1991, now abandoned. The disclosure of
10 the aforementioned U.S. Application No. 09/619,000 (hereinafter, "Referenced Application")
is hereby incorporated by reference as if fully set forth herein.

BACKGROUND OF THE INVENTION

15 Present day hunger for data and data storage has given rise to computing
complexes in which multiple data processing systems have access to a single data warehouse
that is often implemented by a complex of disk storage units. The growth of Internet use has
fed this hunger, and added the requirement that the data be continuously available. In order
to achieve this latter requirement, many database complexes and data warehouses resort to
20 such techniques as "mirroring" (i.e. using redundant storage to maintain a copy of everything
written to the main storage element), error correction of various types and the like.
Redundant arrays of independent (or inexpensive) disks (RAID) is one example. Certain
RAID configurations ("levels") use data striping (spreading out blocks of each file across
multiple disks) in order to protect the data, correcting errors when encounter, but redundancy
25 is not used. This improves performance, but does not deliver fault tolerance. Other RAID
levels (e.g., level 1) provide disk mirroring to add data redundancy and thereby fault
tolerance.

While these techniques operate well to provide a measure of fault tolerance
and, therefore, some continuous availability of the stored data, they can be overloaded when
30 facilities employing these techniques must respond to a large volume of requests for data.

Thus, although continuous availability is now a feature of many database
complexes and data warehouse configurations, they still can present a performance

impediment by limiting the number of accesses that can be made at any particular moment in time.

SUMMARY OF THE INVENTION

5

The present invention is directed to a disk storage system with a storage control unit capable of receiving and simultaneously responding to multiple input/output (I/O) read requests from multiple users of the storage system.

10 Broadly, the invention is a disk storage system in which the storage control unit operates to control data transfers (i.e., reads and writes) between a number of host systems and a physical storage formed by a number of disk storage units. The storage control unit is preferably constructed to include multiple processor units (i.e., microprocessors), providing a platform that allows multiple processes to handle a number of simultaneous data transfers between the physical storage and the host systems. The control unit includes
15 memory in which are maintained data structures that implement "logical" storage, comprising a number of logical storage units to which I/O requests, both reads and writes, are made by the host systems. Each logical storage unit has a designated corresponding physical storage area in the physical storage. Data is written to a predetermined one of the logical storage units (the "master" logical unit), and to its corresponding physical storage area. That data is
20 also copied to the other logical storage units (the "slave" logical storage units), and through them to their corresponding physical storage areas. Thereby, multiple copies of the data is made available.

According to one embodiment of the invention, I/O read requests from the host systems are received and assigned to one of the logical storage units and, thereby, to the
25 corresponding physical disk storage. Selection of a logical storage unit for assignment of an I/O read request is preferably made in a manner that distributes I/O read requests among the logical storage units and, thereby, the corresponding physical storage areas. For example, selection may be made on a round-robin basis, or any other basis that achieves a desired distribution among the logical storage units. In this manner, read requests are distributed
30 over the areas of physical storage containing the multiple copies of the data requested.

In an alternate embodiment of the invention, I/O read requests are not immediately assigned to a logical unit. Rather, all the logical units are mapped to their matching physical discs maintaining the copies of the data of the I/O read request, and those physical disk storage areas examined for selection. For example, the physical disk with the

smallest number of pending requests may be selected, and the I/O read request is assigned to the logical storage unit corresponding to that selected physical storage.

In a further embodiment of the invention, when an I/O read request is received, a number of the logical units, less than all, are mapped to their corresponding physical storage containing the requested data. The physical disks making up that physical storage are then reviewed to select, for example, the one with a small backlog of pending requests, and the I/O read request is assigned to the logical storage unit corresponding to the selected physical storage.

A number of advantages should now be evident to those skilled in this art. Rather than forming a bottleneck by having only a single data stream from a storage facility to multiple hosts, the storage system of the present invention provides multiple, parallel data paths between multiple processors and physical storage. This, in turn, provides almost instantaneous access to data. In addition, fault tolerance and continuous availability is provided.

These, and other advantages and aspects of the invention will become apparent to those skilled in the art upon reading of the following description of the specific embodiments, which should be taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a simplified block diagram illustrating the storage control system of the present invention for distributing I/O read requests from a number of host processor systems over multiple copies of data maintained by physical storage;

Fig. 2 is a diagram of the logical disk group table, a one of the data structures used to implement a group of logical disk storage devices;

Fig. 3 is a logical disk table, identifying, for the corresponding logical storage device, the particulars of that device;

Fig. 4 shows the entry containing the processing request number for a physical disk of Fig. 1, indicating the backlog of I/O requests pending on the corresponding physical disk;

Figs 5-8 are flow diagrams illustrating the steps taken by various of the processes used to respond to an I/O read request to transfer requested data from physical storage to the host system requesting that data;

Fig. 9 is a flow diagram showing the steps taken, according to one embodiment of the invention, to distribute read requests over the physical storage of Fig. 1;

Fig. 10 is a flow diagram showing the steps taken, according to another embodiment of the invention, to distribute read requests over the physical storage of Fig. 1;

and

Fig. 11 is a flow diagram showing the steps taken, according to yet another embodiment of the invention, to distribute read requests over the physical storage of Fig. 1.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

Turning now to the Figures, and for the moment specifically Fig. 1, there is illustrated a storage system, designated generally with the reference numeral 10 for handling I/O data communication between a number of host systems 12 ($12_1, 12_2, \dots, 12_k$) and a physical storage 14, comprising disk units 16 ($16_1, 16_2, \dots, 16_n$). The storage system 10 includes a controller unit 20, with associated memory/cache unit 22, operating to control data transfers between the host system 12 and the physical storage 14 (i.e., disk storage units 16). The controller 20 is preferably implemented by a number of individual processor elements (e.g., microprocessors) or similar devices in order to provide parallel processing, so that multiple different processes can be running simultaneously. For example, the controller 20 may utilize a symmetric multiprocessing (SMP) architecture to exploit the power of multiple processor units through distribution of the operating system to thereby balance the system load of the SMP configuration. In addition, an SMP configuration can reduce downtime because the operating system can continue to run on remaining processor units in the event one the processor units should fail. Alternatively, a single processing element may be used in time-share fashion to provide virtual parallel processing.

Maintained in the memory/cache unit 22 are data structures that implement representations of logical storage "coupling groups," each coupling group comprising one or more logical volumes. While there can be any practical number of logical coupling groups, only one, coupling group 30, is shown in Fig. 1 for the sake of simplicity. However, it should be understood that the discussion of the coupling group will apply to any other coupling group that may be represented by data structures in the memory/cache unit 22.

As Fig. 1 illustrates, a logical coupling group 30 comprises logical volumes 32 ($32_1, 32_2, \dots, 32_m$), the number of which will typically be less than or equal to the number of

physical disk units 16, i.e., $m \leq n$. Logical volume addresses map to addresses of the physical storage 14. As will be seen, a logical volume 32 can map, on a one-to-one basis, to one or more of the disk units 16 of the physical storage 14, or to a portion of a disk unit 16.

In this embodiment, one of the logical volumes 32₁ is designated the “master”
5 volume. All I/O data read or write requests made by a host system 12 are assumed to be directed to the master logical volume 32₁. The remaining logical volumes 32 (32₂, ..., 32_m) are designated “slave” volumes.

I/O read or write requests that issue from the host systems 12 will include an address at which data is to be written (if an I/O write request) or from which data is to be read
10 (if an I/O read request). I/O write requests use the address contained in the request to map to a corresponding address of physical storage 14, and the data then is written to the master logical volume 32₁. In addition, the data of the received I/O write request is also passed to each of the logical volumes 32₂, ..., 32_m of the logical coupling group 30, and similarly mapped and written to the areas of physical storage 14 corresponding to those logical slave
15 volumes. Thus, multiple copies of data written to the storage system 10 are maintained and made available by the physical storage 14 for use by the present invention.

I/O read requests are processed, according to the present invention, in a manner that allows multiple requests to be handled in parallel. A received I/O read request is ultimately assigned to one of the logical volumes 32 for retrieval of the requested data from
20 the physical storage corresponding to that logical slave volume. The present invention provides various methods for distributing the read requests among the logical volumes 32 for parallel handling.

There are three variations of load distribution according to the present invention. An initial load distribution method will first assign an incoming I/O read request
25 to one logical volume 32. For example, a round-robin algorithm may be used for such assignment, although it will be evident to those skilled in this art that other selection methods may be employed. The address of the data contained in the I/O read request will be mapped from the selected logical volume 32 to an address within its corresponding physical storage. The data requested by the I/O read request will then be read from the physical disk 16 of the
30 physical storage corresponding to the selected logical volume, and returned to the host system 12 that originated the request.

In a second variation, the method of the present invention will first perform the logical-physical mapping of all logical volumes 32 to obtain all the addresses of the data

sought by the request in the physical storage 14. Then, a determination is made of which physical disk unit should be selected. For example, that selection may be based upon how many outstanding requests each physical disk has, and the I/O read request then assigned to the logical volume 32 corresponding to that selected disk unit 16 .

5 Finally, according to the third variation of the present invention, an incoming I/O read request will again initiate a logical-physical mapping in order to obtain corresponding physical addresses of the data sought by the request in the physical storage 14. This time, however, less than the maximum number of logical volumes 32, for example only two of the logical slave volumes, are mapped to their corresponding physical storage. Again,
10 as with the second variation,(for example, that selection may be based upon how many outstanding requests each physical disk has), a physical disk is selected, and the I/O read request assigned to the logical slave volume 32 corresponding to such selected physical disk 16.

 Figs. 2 and 3 illustrate the data structures used to establish the representative
15 logical volumes 32. Turning first to Fig. 2, a logical coupling group table 40 is shown. As indicated above, there can be a number of logical groups (the coupling groups), although only one, logical group 30, is shown in Fig. 1. There would be, however, a separate logical coupling group table 40 for each the logical coupling group to be implemented. The logical coupling group table 40 shown in Fig. 2 corresponds to the coupling group 30 of Fig. 1. As
20 Fig. 2 shows, a first entry 42 identifies the number of logical volumes 32 in the particular coupling group. The next entry 44 identifies the address of the master logical volume 32₁. Entry 44 is followed by a number of entries 46 (46₂, ..., 46_m), each containing the address of a corresponding slave volume 32₂, ..., 32_m. When an I/O read request is received, the address contained in the request will be the address of the target master volume 32₁. If,
25 however, there are more than one logical coupling groups, the coupling group tables 40 corresponding to each of those coupling groups will be searched with target address of a received I/O read request to find the master logical volume identified in the request.

 Referring now to Fig. 3, there is shown a logical volume table 50 with information describing the makeup of a logical volume. There will a logical volume table 50
30 for each of the logical volumes 32. A first entry 52 of the logical volume table 50 identifies the mapping type used for storing data of the logical volume on its corresponding physical storage. For example, the mapping type may be a RAID 1 or a RAID 5 mapping, or any other mapping type. Entry 54 identifies the total capacity of the logical volume, as

implemented by its corresponding physical storage 14, and entry 56 identifies the number of physical disk units that implement the logical volume 32 to which the table 50 corresponds.

Following the entry 56 are one or more entries 60. There will be one entry 60 for each of the number of physical disks that correspond to the logical volume of the table 50.

5 Each of the entries 60 will have three fields 64, 66, and 68, respectively identifying the address of the disk unit 16 used by the logical volume, the start, and end, of the physical block to which the corresponding logical volume is mapped.

Fig. 4 shows a physical disk table 70. There will be a physical disk table 70 for each one of the physical disk units 16. The physical disk table 70 has only one entry 72,
10 identifying the number of requests then pending and/or then being processed by the disk unit 16 to which the table 70 pertains.

Turning now to Figs. 5-8, illustrated are the four processes, running on the multiple microprocessors that make up the controller 20, for handling the transfer of data in response to read requests issued by the host systems 12. There are a number of sets of the
15 processes shown in Figs. 5-8 to make available parallel data transfer operations between the processor systems 12 and the physical storage 14.

Turning first to Fig. 5, the flow diagram 40 for a read request process used in connection with the present invention is illustrated. The read request process operates to receive an I/O read request from a host system 12, determine whether the requested data may
20 be in the memory/cache unit 22 or whether physical storage 14 must be accessed to retrieve the data, and pass the request to the appropriate process depending upon the determination made. As Fig. 5 shows, the read request process will receive, at step 82, the I/O read request. It will then, in step 84, make a determination of the location of the requested data, i.e., whether contained in the memory/cache unit 22 or not in which case it must be retrieved from
25 physical storage 14). If so, step 84 is left in favor of step 86 in which the request will be passed to the host read transfer process 90 (Fig. 6), and terminate at step 88. If, however, step 24 determined that the requested data is not located in the memory/cache unit 22, step 84 will be left in favor of step 87 in which the I/O read request is passed to a load distribution process, described more fully below, for accessing physical storage 14 (i.e., one of the disk
30 units 16) for the data.

Assuming that the requested data is in the memory/cache unit 22, step 86 of the read request process 80 will pass the request to the host read transfer process, illustrated by the flow diagram 90 in Fig. 6. There, at step 92, the host read transfer process will execute

a transfer of the data to the requesting host by reading the data from the memory/cache unit 22, and transfer it to the requesting host system 12, terminating at step 94.

On the other hand, should the requested data be found in step 84 of the read request process to not be contained in the memory/cache unit 22, the controller 20 will invoke a load distribution process to select one of the logical units 32, and thereby a corresponding physical disk unit 16, to read the data. When a physical disk unit is finally selected, the read request disk issue process, illustrated by the flow diagram 100 shown in Fig. 7, is employed. The read request disk issue process begins with step 102 by incrementing the processing request number contained in the entry 72 of the physical disk table 70 (Fig. 4) to indicate that the selected disk unit 16 is processing an additional request. Step 102 of the read request disk issue process is then followed by step 104 in which the read request is issued to the selected physical disk 16. The read request disk issue process then terminates with step 106.

The selected disk unit 16 receiving the read request sent in step 104 of the read request disk issue process 100 will respond with the necessary preparations for reading the requested data and issue an interrupt to the controller 20 to indicate that the data is now ready for transfer, as is conventional. This will invoke the disk read transfer process, the flow diagram 110 of which is shown in Fig. 8. The disk read transfer process, upon receipt of the interrupt indicating data ready by a disk unit 16, will, at step 112, transfer the requested data from the selected disk unit 16 to the memory/cache unit 22. At step 114, the entry 70 containing the processing request 72 is decrement to indicate that the particular disk unit 16 has completed a pending request. Then, in step 116, the disk read transfer process 110 will transfer the data from the memory/cache unit 22 to the requesting host processor system 12, and terminate at step 118.

As indicated above, data is written by a host system 12 to storage system 10 by addressing the master logical unit 32₁. That data is then mapped from the master logical volume 32₁ to its corresponding physical storage 14, i.e., a disk unit or disk units 16. In addition, as also indicated above, that data is copied to each of the slave logical units 32₂, ... 32_m of the coupling group 30, and thereby to the corresponding disk units 16. Thus, multiple copies of data written to the storage system 10 provide fault tolerance and continuous data availability. In addition, the availability of multiple copies forms the basis of the load distribution methods of the present invention. I/O read requests that are received are distributed among the logical units 32 of the logical coupling group 30. A first embodiment of the load distribution process 120, according to one embodiment of the present invention, is broadly illustrated by the flow chart of Fig. 9.

The load distribution process 120 is called by step 87 of the read request process 80 (Fig. 5) when the data of an I/O read request is not found in the memory/cache unit 22. When called, the load distribution process 120 begins at step 122 by assigning the I/O read request to one of the logical units 32 ($32_1, \dots, 32_m$). The assignment can be any method of selection or arbitration such as prearranged priorities as to which logical slave unit received a next I/O read request or other classic arbitration techniques or a "round-robin" technique. The round-robin technique involves assigning priorities according to a predetermined order among the logical units 32. However determined, the load distribution process 120 will then move to step 124 to execute a logical/physical mapping of the selected logical unit 32 to its corresponding physical disk unit 16, employing the information of the logical volume table 50 (Fig. 3).

The load distribution process of flow diagram 120, having located the data in physical storage 14, will then, in step 126, pass the I/O read request to the read request disk issue process 100 (Fig. 7) which will ultimately result in the requested data being returned to the requesting processor system 12 as described above.

The load distribution process 120 then ends at step 128.

Fig. 10 illustrates, in flow diagram form, an alternate load distribution process 130. As before, the load distribution process 130 will be called when it is determined, by step 84 of the read request process 80 (Fig. 5) that the data of an I/O read request is not located in the memory/cache unit 22. Accordingly, step 84 is left in favor of step 87 which will pass the I/O read request to the load distribution process 130 where, at step 132, a logical/physical mapping of all logical units 32 ($32_1, \dots, 32_m$) of the logical coupling group 30 is performed. This will obtain the addresses of the physical storage 14, corresponding to each of the logical units 32, where at the requested data resides.

With the physical storage addresses now available, the load distribution process 130, at step 134, will then, for each of the disk units 16, check the corresponding entry 70, examining the processing request 72 for each. Then, in step 135, the load distribution process 130 will select that physical disk unit having the lowest processing request 72 and allocate the I/O read request.

With the physical disk unit 16 so identified, that identification, together with the I/O read request is then passed to the read request disk issue process 100 (Fig. 7) to initiate the transfer of the data from the physical storage 14 to the requesting processor system 12, as described above.

Finally, the load distribution process 130 will terminate at step 138.

A still another embodiment of the load distribution process of the present invention is illustrated in Fig. 11 as load distribution process 150. As shown, the load distribution process 150 begins with step 152 when the I/O read request is passed thereto by step 87 of the read request process 80 (Fig. 5). In step 152, the load distribution process 150 will select one or more of the logical units 32 of coupling group 30. The selected logical units 32 are, in step 154, mapped to their corresponding physical units 16 of physical storage 14.

Then, in step 156, the load distribution process 150 will access the entries 70 corresponding to each of the selected physical disks 16, to obtain each of their processing request 72. In step 160, comparing the processing request 72 for each of the physical disk units 16 to one another, one (e.g., the one with the lowest number) will be selected.

Step 162 checks the processing request 72 of the selected physical disk 16 to determine if the "load" of the selected physical disk is appropriate; that is, if the processing request 72 is within a predetermined parameter (e.g., what ?). If so, step 162 will exit in favor of step 164 in which the load distribution process 150 will pass the I/O read request to the read request disk issue process 100 (Fig. 7) in order to access the data and initiate the steps necessary to pass that data to the requesting processor system 12. The load distribution process 150 will then terminate at step 166.

If, however, step 162 determines that the load condition of the selected physical disk unit 16 is not within an acceptable parameter, step 162 will be exited in favor of determining, in step 170, whether or not a less restrictive load condition is needed. If so, in step 172, the load condition restriction is made more liberal, and the load distribution process 150 returns to step 152 to again locate the data on a physical disk whose load is now within the modified load condition parameter.

Similarly, if it is determined in step 170 that a less restrictive load condition is not needed, step 170 is left to return the load distribution process 150 to step 152 to again select a physical disk.